IFTM - INSTITUTO FEDERAL DO TRIÂNGULO MINEIRO

ENGENHARIA DE COMPUTAÇÃO

Yuri David Silva Duarte

José Ferreira Arantes Lopes

PROJETO FINAL
JOGO DA VELHA UTILIZANDO PIC

Uberaba - MG

2024

SUMÁRIO

1. INTRODUÇÃO

O presente relatório descreve o desenvolvimento e implementação de um projeto prático da disciplina de Microprocessadores e Microcontroladores, focado na criação de um jogo da velha utilizando o microcontrolador PIC16F887. Este projeto teve como objetivo aplicar os conhecimentos adquiridos em sala de aula sobre programação e manuseio de microcontroladores, bem como desenvolver habilidades práticas na construção de circuitos eletrônicos.

Inicialmente, o escopo do projeto previa a utilização de três botões (um para mover o cursor, um para confirmar a posição da jogada e um para resetar o jogo), dois displays de 7 segmentos, o microcontrolador PIC16F887 e um oscilador. No entanto, devido a ajustes durante o desenvolvimento, a versão final do projeto foi realizada utilizando dois botões (um para mover o cursor e outro para confirmar a posição da jogada) e o oscilador, mantendo o PIC16F887 como o núcleo do sistema.

Este relatório detalha todo o processo de desenvolvimento, desde o planejamento inicial e a definição dos componentes até a implementação do código e a montagem do circuito. São discutidos os desafios enfrentados e as soluções adotadas para garantir o funcionamento adequado do jogo da velha, destacando a utilização dos botões para a interação com o jogador e a lógica programada no microcontrolador para gerenciar o jogo.

O desenvolvimento deste projeto proporcionou uma experiência prática valiosa, permitindo aos integrantes do grupo consolidar seus conhecimentos em microprocessadores e microcontroladores, além de desenvolver habilidades de resolução de problemas e trabalho em equipe.

2. MATERIAIS

2.1. Microcontrolador

Como microcontrolador utilizamos o PIC16F887. Suas portas são dispostas conforme mostrado na **Tabela 1** retirada do datasheet (ALLDATASHEET, 2007).

I/O	Pin	Analog	Comparators	Timers	ECCP	EUSART	MSSP	Interrupt	Pull-up	Basic
RA0	2	AN0/ULPWU	C12IN0-	—	—	—	—	—	—	—
RA1	3	AN1	C12IN1-	—	—	—	—	—	—	—
RA2	4	AN2	C2IN+	—	—	—	—	—	—	VREF-/CVREF
RA3	5	AN3	C1IN+	—	—	—	—	—	—	VREF+
RA4	6	—	C1OUT	T0CKI	—	—	—	—	—	—
RA5	7	AN4	C2OUT	—	—	—	$\overline{SS}$	—	—	—
RA6	14	—	—	—	—	—	—	—	—	OSC2/CLKOUT
RA7	13	—	—	—	—	—	—	—	—	OSC1/CLKIN
RB0	33	AN12	—	—	—	—	—	IOC/INT	Y	—
RB1	34	AN10	C12IN3-	—	—	—	—	IOC	Y	—
RB2	35	AN8	—	—	—	—	—	IOC	Y	—
RB3	36	AN9	C12IN2-	—	—	—	—	IOC	Y	PGM
RB4	37	AN11	—	—	—	—	—	IOC	Y	—
RB5	38	AN13	—	$\overline{T1G}$	—	—	—	IOC	Y	—
RB6	39	—	—	—	—	—	—	IOC	Y	ICSPCLK
RB7	40	—	—	—	—	—	—	IOC	Y	ICSPDAT
RC0	15	—	—	T1OSO/T1CKI	—	—	—	—	—	—
RC1	16	—	—	T1OSI	CCP2	—	—	—	—	—
RC2	17	—	—	—	CCP1/P1A	—	—	—	—	—
RC3	18	—	—	—	—	—	SCK/SCL	—	—	—
RC4	23	—	—	—	—	—	SDI/SDA	—	—	—
RC5	24	—	—	—	—	—	SDO	—	—	—
RC6	25	—	—	—	—	TX/CK	—	—	—	—
RC7	26	—	—	—	—	RX/DT	—	—	—	—
RD0	19	—	—	—	—	—	—	—	—	—
RD1	20	—	—	—	—	—	—	—	—	—
RD2	21	—	—	—	—	—	—	—	—	—
RD3	22	—	—	—	—	—	—	—	—	—
RD4	27	—	—	—	—	—	—	—	—	—
RD5	28	—	—	—	P1B	—	—	—	—	—
RD6	29	—	—	—	P1C	—	—	—	—	—
RD7	30	—	—	—	P1D	—	—	—	—	—
RE0	8	AN5	—	—	—	—	—	—	—	—
RE1	9	AN6	—	—	—	—	—	—	—	—
RE2	10	AN7	—	—	—	—	—	—	—	—
RE3	1	—	—	—	—	—	—	—	Y(1)	$\overline{MCLR}$/VPP
—	11	—	—	—	—	—	—	—	—	VDD
—	32	—	—	—	—	—	—	—	—	VDD
—	12	—	—	—	—	—	—	—	—	VSS
—	31	—	—	—	—	—	—	—	—	VSS

Note 1: Pull-up activated only with external $\overline{MCLR}$ configuration.

Tabela 1: Portas de entrada e saída do PIC16F887.

No escopo do projeto estava escrito que utilizaríamos 3 entradas digitais e 23 saídas digitais, sendo 9 para os LEDs e 14 saídas para os displays de 7 segmentos. O que obtemos no final foi a utilização de 2 entradas digitais e 9 saídas digitais, uma vez que não usamos os displays.

2.2. Visualização

O tabuleiro do jogo é mostrado por uma matriz 3x3 feita com LEDs RGB conforme a **Figura 1**. A cor vermelha indica uma posição marcada pelo jogador 1, a cor verde indica que foi marcada pelo jogador 2, a cor laranja representa o cursor e o LED apagado é um local disponível para jogar.

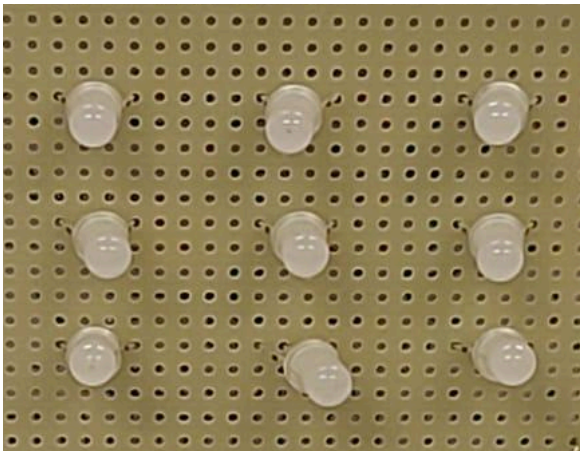


Figura 1: Matriz de LEDs.

2.3. Componentes e custos

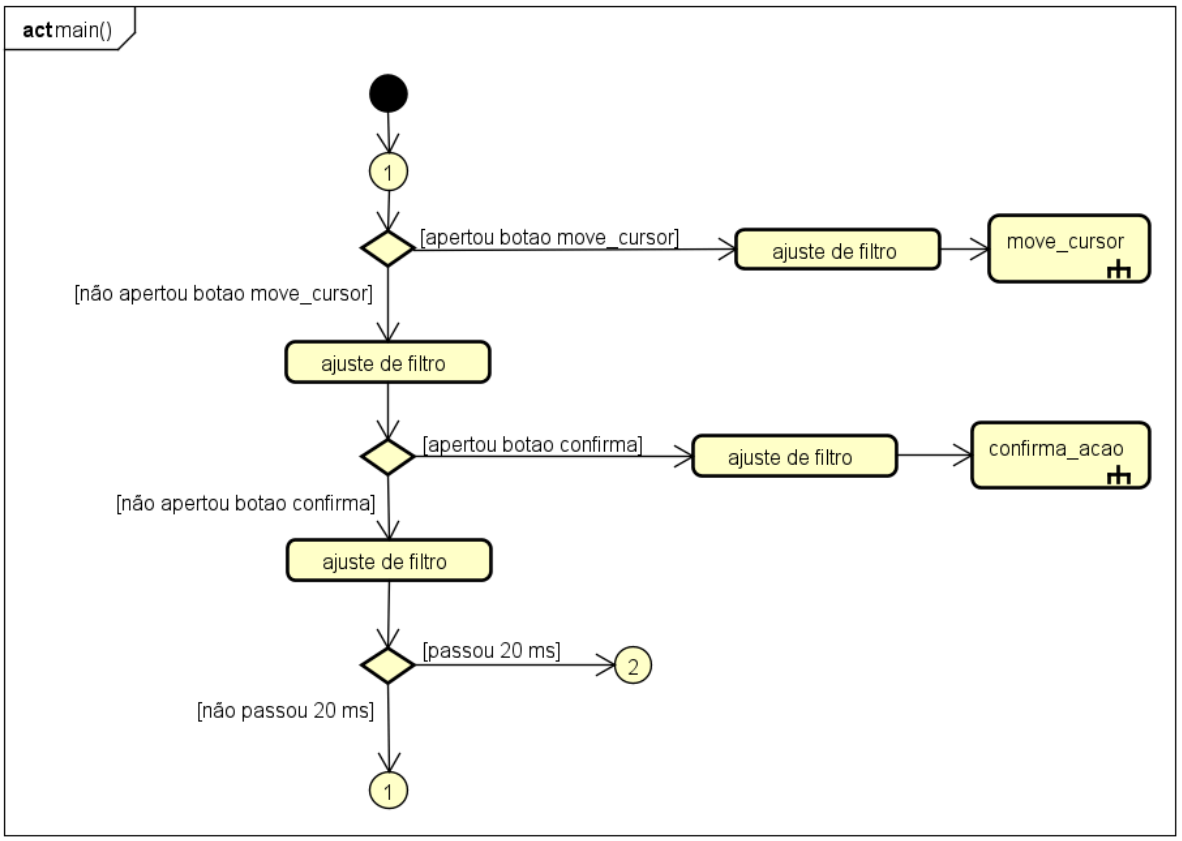
Componente	Quantidade	Custo Unitário	Custo Total
Capacitor de 100 nF	2	R$ 0,15	R$ 0,30
Microcontrolador PIC16F877A	1	R$ 29,29	R$ 29,29
Led RGB Catodo Comum	9	R$ 1,00	R$ 9,00
Transistor Npn BC337	3	R$ 0,50	R$ 1,50
Resistores de 470 Ω	3	R$ 0,08	R$ 0,24
Resistores de 330 Ω	3	R$ 0,08	R$ 0,24
Resistores de 4,7 Kω	5	R$ 0,08	R$ 0,40
Push Buttons	3	R$ 1,15	R$ 3,45
Placa Perfurada	1	R$ 7,90	R$ 7,90
Displays de 7 Segmentos	1	R$ 1,44	R$ 1,44
Resistores de 560Ω	14	R$ 0,08	R$ 1,12
Oscilador de Cristal 8MHz	1	R$ 1,20	R$ 1,20
Capacitores 22pF	2	R$ 0,15	R$ 0,30
Frete Shopee	1	R$ 8,00	R$ 8,00
Frete Eletrogate	1	R$ 6,54	R$ 6,54

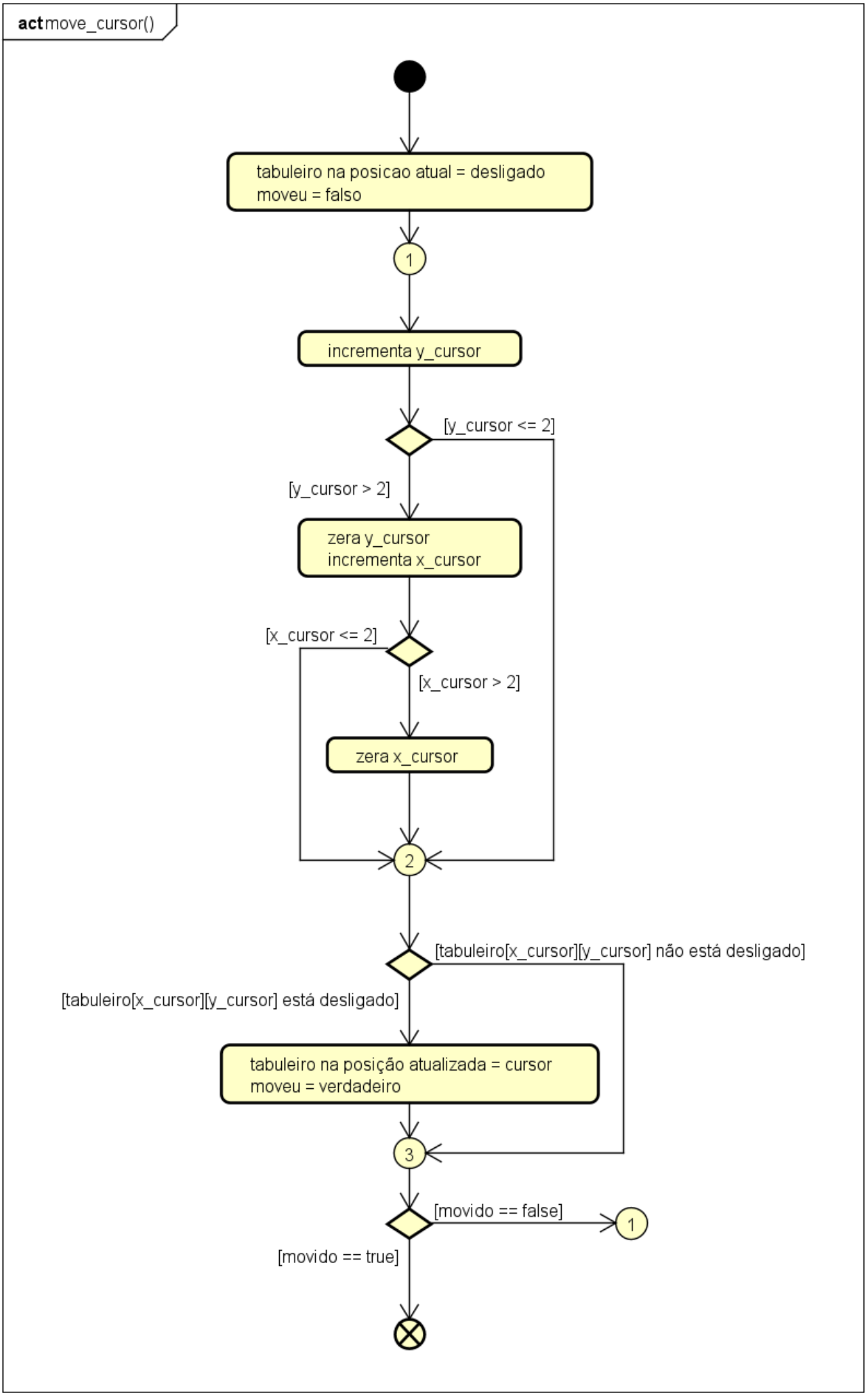
Frete Instituto Digital	1	R$ 11,76	R$ 11,76
Total	36	R$ 45,10	R$ 82,68

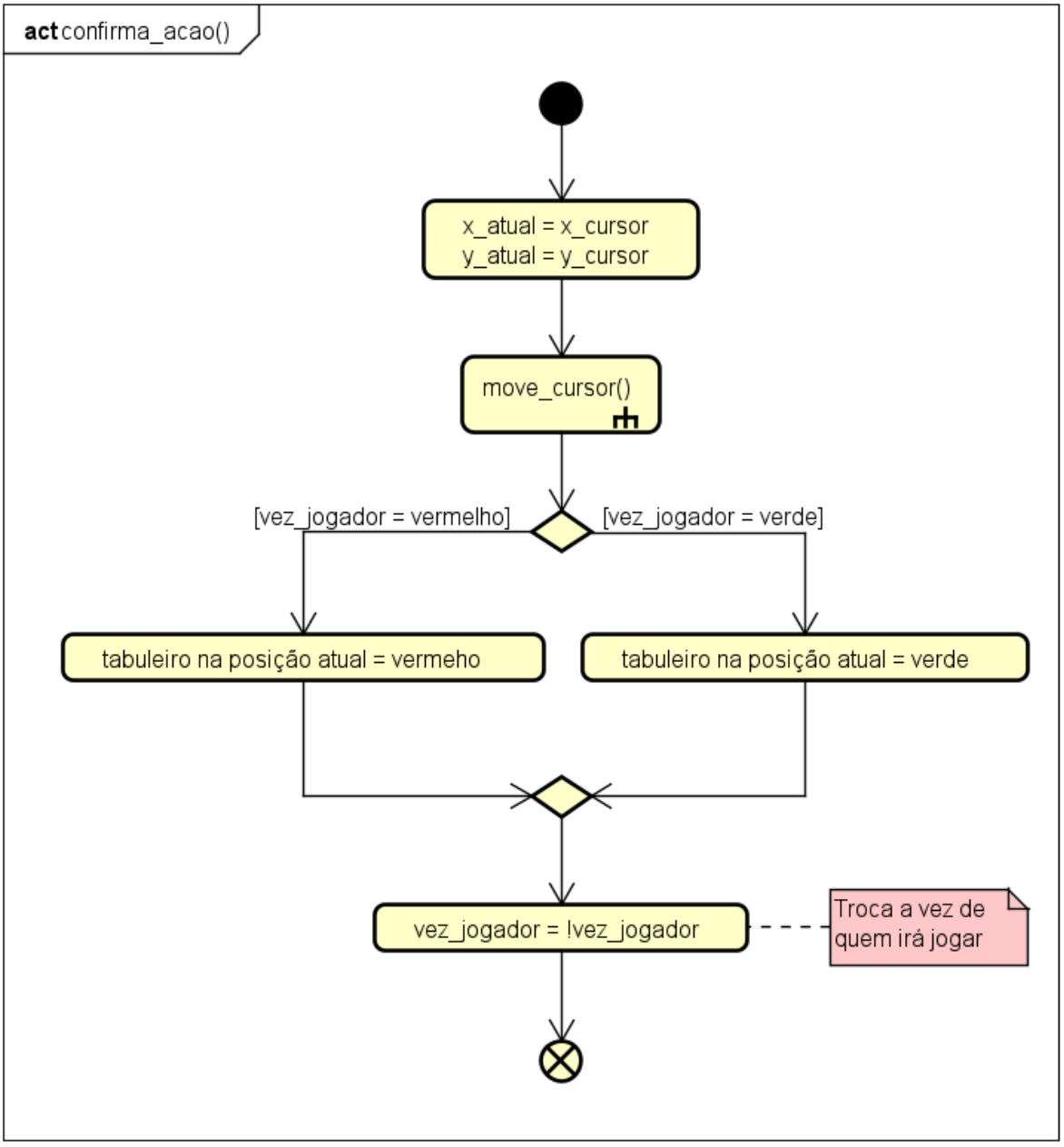
2.4. Aplicações

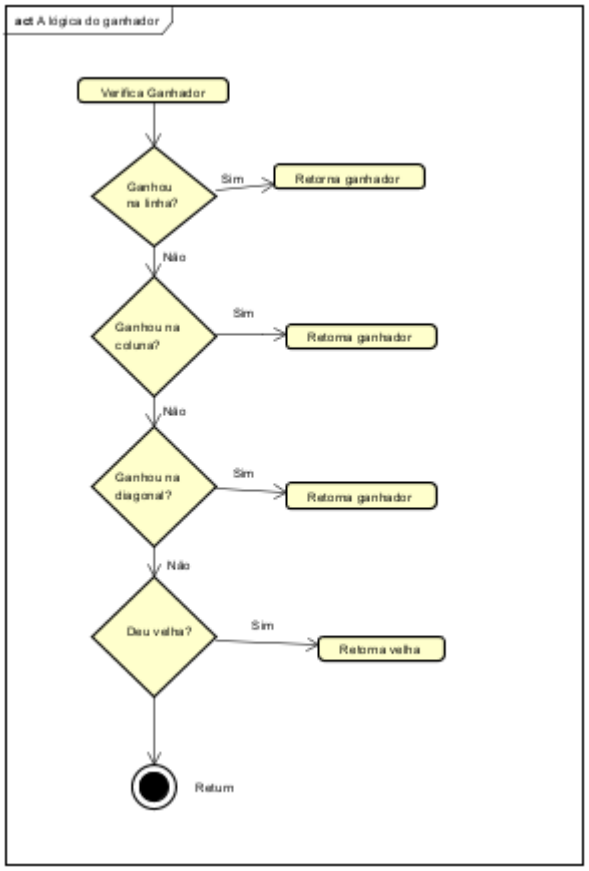
- Na criação:
  - Ganhar conhecimento em sistemas embarcados;
  - Habilidades adquiridas em montagem de circuitos em placa perfurada;
  - Desenvolvimento em interface de usuário.
- Com ele finalizado:
  - Passatempo;
  - Interação com amigos;

2.5. Diagramas Funcionais




act move_cursor()
tabuleiro na posicao atual = desligado
moveu = falso
1
incrementa y_cursor
[y_cursor <= 2]
[y_cursor > 2]
zera y_cursor
incrementa x_cursor
[x_cursor <= 2]
[x_cursor > 2]
zera x_cursor
2
[tabuleiro[x_cursor][y_cursor] não está desligado]
[tabuleiro[x_cursor][y_cursor] está desligado]
tabuleiro na posição atualizada = cursor
moveu = verdadeiro
3
[movido == false]
1
[movido == true]


act confirma_acao()
x_atual = x_cursor
y_atual = y_cursor
move_cursor()
[vez_jogador = vermelho]
[vez_jogador = verde]
tabuleiro na posição atual = vermeho
tabuleiro na posição atual = verde
vez_jogador = !vez_jogador
Troca a vez de quem irá jogar



2.6. Código fonte

ProjetoFinal.c

```
#include <PF.h>

int1 fimtempo = 0, ja_li_reset = 0, ja_li_mov = 0, ja_li_conf = 0,
isMovido = 0, vez_jogador_verde = 1;
int16 tempo = 0;
int8 i = 0, j = 0, x = 0, y = 0, cursorx = 0, cursory = 0,
filtro_reset = 100, filtro_mov = 100, filtro_conf = 100;
int8 pontuacaoJ1 = 0, pontuacaoJ2 = 0, ganhador = 0, contvelha =
0, x_aux, y_aux;
int8 tabuleiro[3][3];

void reinicia_tabuleiro();
void acende_tabuleiro();
void move_cursor();
int8 verifica_ganhador(int8 matriz[3][3]);
void zera_jogo();
void confirma_acao();
```

```
void atualiza_displays();
int8 converte_para_display(int8);

#INT_TIMER0
void TIMER0_isr(void)
{
    tempo++;

    if (tempo == 20)
    {
        tempo = 0;
        fimtempo = 1;
    }
}

void main()
{
    setup_timer_0(RTCC_INTERNAL | RTCC_DIV_8 | RTCC_8_BIT); // 1,
0 ms overflow
    enable_interrupts(INT_TIMER0);
    enable_interrupts(GLOBAL);

    zera_jogo();
//!    atualiza_displays();

    while (TRUE)
    {
//!        if (input(B_RESET) == 0)
//!        {
//!            filtro_reset--;
//!            if (filtro_reset == 0 && ja_li_reset == 0)
//!            {
//!                ja_li_reset = 1;
//!                zera_jogo();
//!            }
//!        }
//!        else
//!        {
//!            ja_li_reset = 0;
//!            filtro_reset = 100;
//!        }
```

```
        if (input(B_MOVE) == 0)
        {
            filtro_mov--;
            if (filtro_mov == 0 && ja_li_mov == 0)
            {
                ja_li_mov = 1;
                move_cursor();
            }
        }
        else
        {
            ja_li_mov = 0;
            filtro_mov = 100;
        }

        if (input(B_CONFIRM) == 0)
        {
            filtro_conf--;
            if (filtro_conf == 0 && ja_li_conf == 0)
            {
                ja_li_conf = 1;
                confirma_acao();
            }
        }
        else
        {
            ja_li_conf = 0;
            filtro_conf = 100;
        }

        if (fimtempo)
        {
            acende_tabuleiro();
            y += 1;
            if (y > 2)
            {
                x += 1;
                y = 0;
                if (x > 2)
                {
                    x = 0;
                }
```

```
            }

//!           atualiza_displays();

            ganhador = verifica_ganhador(tabuleiro);

            if (ganhador != 0)
            {
                if (ganhador == 1)
                {
                    pontuacaoJ1++;
                }
                else if (ganhador == 2)
                {
                    pontuacaoJ2++;
                }
                else if (ganhador == 3)
                {
                    // DAR UM RETORNO VISUAL PARA O USUARIO QUE
DEU VELHA
                }

                reinicia_tabuleiro();
            }

//!           atualiza_displays();

            fimtempo = 0;
        }
    }
}

void atualiza_displays()
{
    output_c(converte_para_display(pontuacaoJ1));

    output_d(converte_para_display(pontuacaoJ2));
}

int8 converte_para_display(valor)
{
    switch (valor)
```

```
    {
    case 0:
        return 64;
        break;
    case 1:
        return 121;
        break;
    case 2:
        return 36;
        break;
    case 3:
        return 48;
        break;
    case 4:
        return 25;
        break;
    case 5:
        return 18;
        break;
    case 6:
        return 2;
        break;
    case 7:
        return 120;
        break;
    case 8:
        return 0;
        break;
    case 9:
        return 16;
        break;
    }
}

void confirma_acao()
{
    x_aux = cursorx;
    y_aux = cursory;

    move_cursor();

    if (vez_jogador_verde == 1)
```

```
    {
        tabuleiro[x_aux][y_aux] = 1;
    }
    else
    {
        tabuleiro[x_aux][y_aux] = 2;
    }

    vez_jogador_verde = !vez_jogador_verde;
}

void reinicia_tabuleiro()
{
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            tabuleiro[i][j] = -1;
        }
    }

    tabuleiro[0][0] = 0;
    cursorx = cursory = 0;

    ganhador = 0;
}

void acende_tabuleiro()
{
    output_high(LIN1);
    output_high(LIN2);
    output_high(LIN3);
    output_low(RED1);
    output_low(RED2);
    output_low(RED3);
    output_low(GREEN1);
    output_low(GREEN2);
    output_low(GREEN3);

    if (x == 0)
    {
        output_low(LIN1);
```

```
    }
    else if (x == 1)
    {
        output_low(LIN2);
    }
    else if (x == 2)
    {
        output_low(LIN3);
    }

    if (tabuleiro[x][y] == 1)
    {
        if (y == 0)
        {
            output_high(RED1);
        }
        else if (y == 1)
        {
            output_high(RED2);
        }
        else
        {
            output_high(RED3);
        }
    }
    else if (tabuleiro[x][y] == 2)
    {
        if (y == 0)
        {
            output_high(GREEN1);
        }
        else if (y == 1)
        {
            output_high(GREEN2);
        }
        else
        {
            output_high(GREEN3);
        }
    }
    else if (tabuleiro[x][y] == 0)
    {
```

```
        if (y == 0)
        {
            output_high(RED1);
            output_high(GREEN1);
        }
        else if (y == 1)
        {
            output_high(RED2);
            output_high(GREEN2);
        }
        else
        {
            output_high(RED3);
            output_high(GREEN3);
        }
    }
}

void move_cursor()
{
    tabuleiro[cursorx][cursory] = -1;
    isMovido = 0;
    while (isMovido == 0)
    {
        cursory++;
        if (cursory > 2)
        {
            cursory = 0;
            cursorx++;
            if (cursorx > 2)
            {
                cursorx = 0;
            }
        }

        if (tabuleiro[cursorx][cursory] == -1)
        {
            tabuleiro[cursorx][cursory] = 0;
            isMovido = 1;
        }
    }
}
```

```
int8 verifica_ganhador(int8 matriz[3][3])
{
    for (i = 0; i < 3; i++)
    {
        if (matriz[i][0] != -1 && matriz[i][0] == matriz[i][1] &&
matriz[i][0] == matriz[i][2])
            return matriz[i][0];

        if (matriz[0][i] != -1 && matriz[0][i] == matriz[1][i] &&
matriz[0][i] == matriz[2][i])
            return matriz[0][i];
    }

    if (matriz[0][0] != -1 && matriz[0][0] == matriz[1][1] &&
matriz[0][0] == matriz[2][2])
        return matriz[0][0];
    else if (matriz[0][2] != -1 && matriz[0][2] == matriz[1][1] &&
matriz[0][2] == matriz[2][0])
        return matriz[0][2];
    else
    {
        contvelha = 0;

        for (i = 0; i < 3; i++)
        {
            for (j = 0; j < 3; j++)
            {
                if (tabuleiro[i][j] != -1 && tabuleiro[i][j] != 0)
                {
                    contvelha++;
                }
            }
        }

        if (contvelha == 9)
            return 3;
        else
        {
            return 0;
        }
    }
```

```
}

void zera_jogo()
{
    reinicia_tabuleiro();

    fimtempo = 0;
    tempo = 0;
    i = j = x = y = 0;
    pontuacaoJ1 = pontuacaoJ2 = contvelha = 0;
}
```

ProjetoFinal.h

```
#include <16F887.h>
#device ADC = 10

#FUSES PUT         // Power Up Timer
#FUSES NOMCLR      // Master Clear pin used for I/O
#FUSES NOPROTECT   // Code not protected from reading
#FUSES NOCPD       // No EE protection
#FUSES NOBROWNOUT // No brownout reset
#FUSES IESO        // Internal External Switch Over mode enabled
#FUSES FCMEN       // Fail-safe clock monitor enabled
#FUSES NOLVP       // No low voltage prgming, B3(PIC16) or
B5(PIC18) used for I/O
#FUSES BORV40      // Brownout reset at 4.0V
#FUSES NOWRT       // Program memory not write protected

#use delay(crystal = 8MHz)
#use FIXED_IO(A_outputs = PIN_A5)
#use FIXED_IO(B_outputs = PIN_B7, PIN_B6, PIN_B5, PIN_B4, PIN_B3,
PIN_B2, PIN_B1, PIN_B0)
#use FIXED_IO(C_outputs = PIN_C7, PIN_C6, PIN_C5, PIN_C4, PIN_C3,
PIN_C2, PIN_C1, PIN_C0)
#use FIXED_IO(D_outputs = PIN_D7, PIN_D6, PIN_D5, PIN_D4, PIN_D3,
PIN_D2, PIN_D1, PIN_D0)
#use FIXED_IO(E_outputs = PIN_E0)
```

```
#define B_CONFIRM PIN_A0
#define B_MOVE PIN_A1
#define B_RESET PIN_A2
#define RED1 PIN_A5
#define RED2 PIN_B0
#define RED3 PIN_B1
#define GREEN1 PIN_B2
#define GREEN2 PIN_B3
#define GREEN3 PIN_B4
#define LIN1 PIN_B7
#define LIN2 PIN_B6
#define LIN3 PIN_B5
#define A1 PIN_C0
#define A2 PIN_C1
#define A3 PIN_C2
#define A4 PIN_C3
#define A5 PIN_C4
#define A6 PIN_C5
#define A7 PIN_C6
#define A8 PIN_C7
#define B1 PIN_D0
#define B2 PIN_D1
#define B3 PIN_D2
#define B4 PIN_D3
#define B5 PIN_D4
#define B6 PIN_D5
#define B7 PIN_D6
#define B8 PIN_D7
```

3. DESENVOLVIMENTO

Para a realização deste projeto, utilizamos como referência inicial o projeto de jogo da velha encontrado no site MakerHero. No entanto, o código completo foi refeito do zero e pode ser acessado em nosso repositório no GitHub (DUARTE; LOPES, 2024).

3.1. Primeiros Passos

O primeiro passo foi a elaboração do escopo do projeto. Com base nele, adquirimos os componentes necessários para a montagem. Quando as peças chegaram, percebemos que o microcontrolador recebido estava incorreto; pedimos o PIC16F877A, mas recebemos o PIC16F887. Embora não tenham sido necessárias muitas adaptações, foi importante trocar a biblioteca de um PIC para o outro.

Enquanto aguardávamos a chegada das peças, começamos a desenvolver o código responsável pela lógica do jogo e a comunicação com o hardware, utilizando o software Proteus 8 Professional para prototipação e testes que pode ser conferido na **Figura 2**. Durante essa fase, notamos a falta de um soquete para fixar o PIC na placa perfurada, o que nos obrigou a buscar uma solução rapidamente.

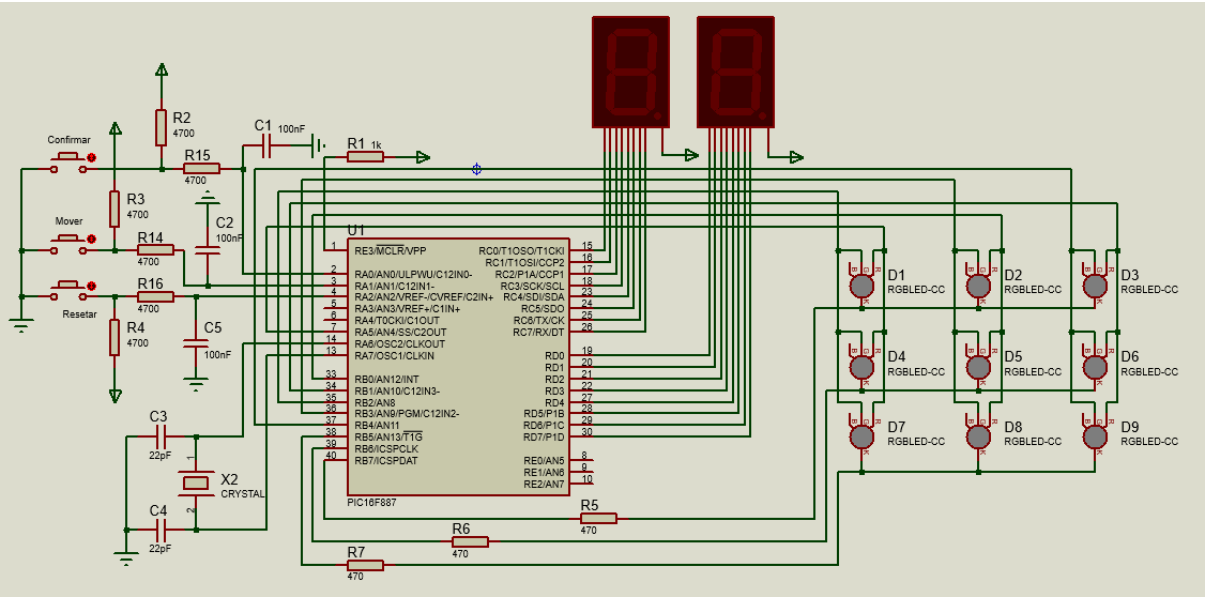


Figura 2: Prototipação com Proteus.

3.2. Programação

A programação foi realizada de forma incremental. Primeiro, desenvolvemos uma função, depois a testamos no Proteus e, em seguida, passamos para a próxima função. Esse método garantiu que cada parte do código funcionasse

corretamente antes de prosseguir, evitando o acúmulo de erros e facilitando a identificação de problemas.

3.3. Montagem e Testes

A montagem e os testes do hardware foram realizados passo a passo. Inicialmente, conectamos os LEDs dispostos em matrizes à placa perfurada, de acordo com o desenho na **Figura 3**, e soldadas às portas corretas do PIC. Em seguida, conectamos e testamos o botão de mover, repetindo o processo para o botão de confirmar jogada.

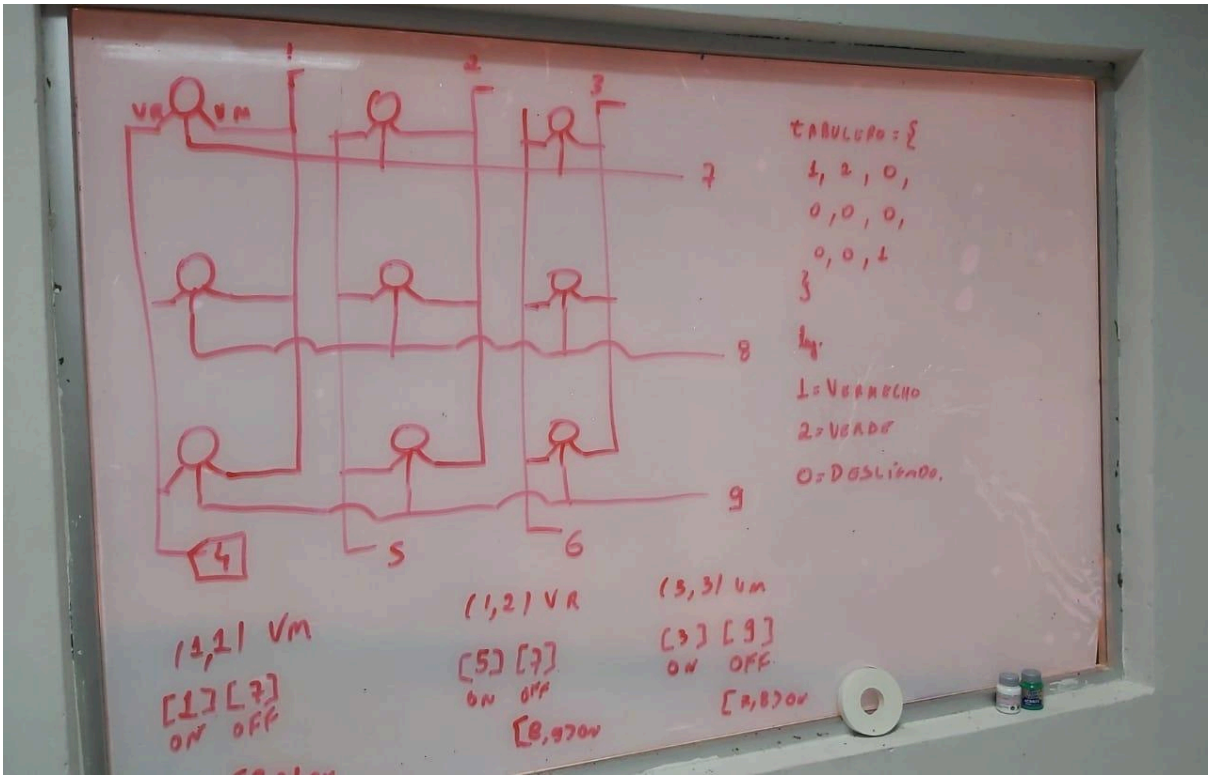


Figura 3: Desenho esquemático da matriz de LEDs.

Durante o desenvolvimento do hardware, a maior dificuldade encontrada foi a soldagem precisa de todos os componentes (**Figura 4**). Esse processo exige muita precisão, foco e habilidade manual. Muitas vezes, dois integrantes eram necessários para realizar a soldagem, o que evidenciou o tempo despendido nessa etapa. Além disso, a disposição dos componentes exigiu cautela e atenção para evitar erros que comprometessem o funcionamento do circuito (**Figura 5**).

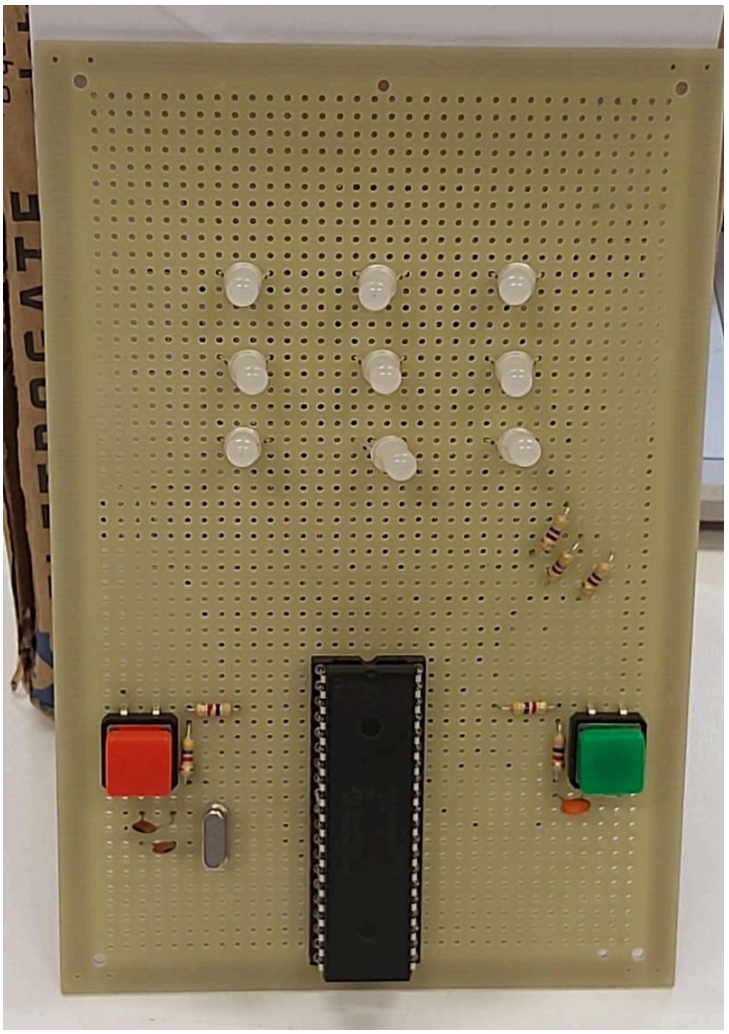


Figura 4: Componentes soldados à placa perfurada.

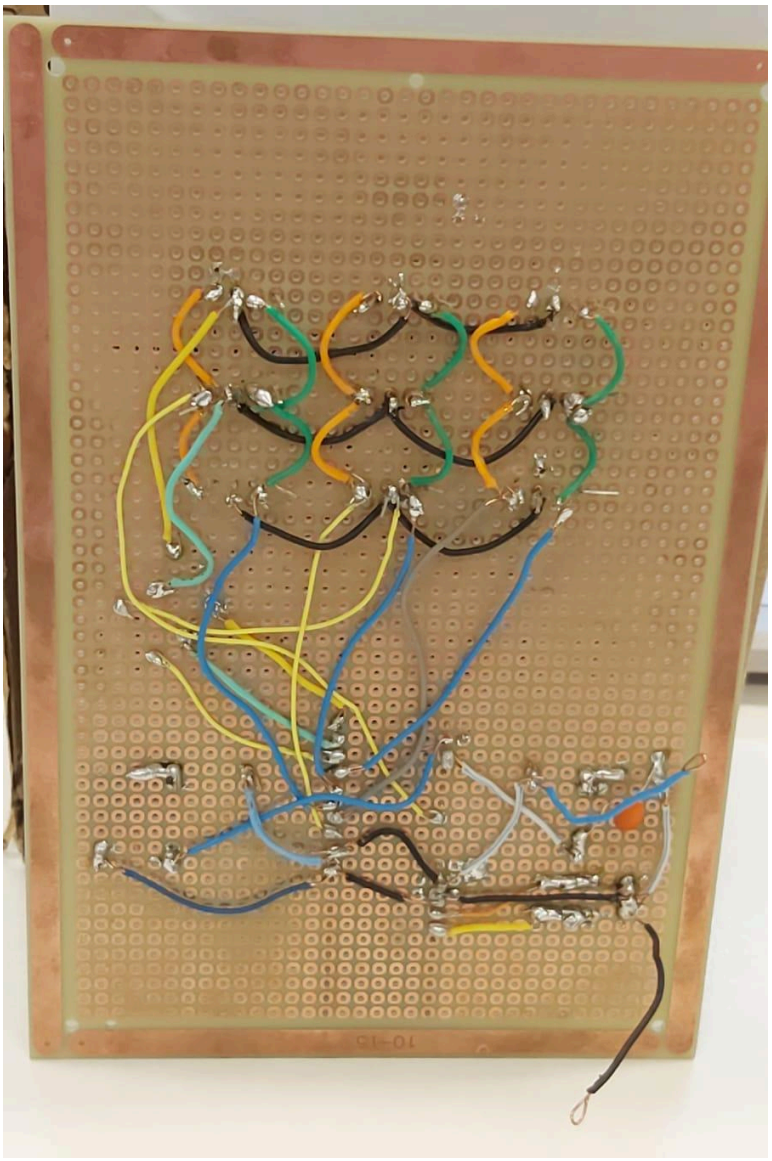


Figura 5: Placa vista por trás.

Infelizmente, não conseguimos conectar os displays que mostrariam a pontuação dos jogadores devido ao curto intervalo de tempo dedicado ao projeto. Por isso, não implementamos o botão de reiniciar, já que sua função seria reiniciar o jogo e as pontuações dos jogadores. Nosso projeto já reinicia o jogo automaticamente a cada vitória ou empate, e a pontuação não foi apresentada.

3.4. Desafios e Soluções

Os principais desafios enfrentados incluíram a adaptação do microcontrolador, a falta de componentes específicos e a soldagem precisa dos componentes na placa perfurada. As soluções adotadas envolveram a troca de bibliotecas, a busca rápida por componentes alternativos e o trabalho colaborativo para a soldagem.

3.5. Avaliação e Próximos Passos

Apesar das dificuldades, o projeto alcançou seus principais objetivos, proporcionando um jogo da velha funcional. Planejamos continuar o desenvolvimento do projeto após o término da disciplina, visando implementar as funcionalidades restantes e aprimorar nossas habilidades em montagem de circuitos e programação de componentes.

4. CONCLUSÃO

O presente trabalho alcançou a maioria dos resultados desejados, especialmente na essência do projeto, que é o jogo da velha. Conseguimos implementar com sucesso a movimentação do cursor, a marcação da posição desejada, a resposta visual de onde foi jogada a peça, a alternância de turnos entre os jogadores e a lógica para determinar vitórias ou empates. No entanto, a apresentação das pontuações dos jogadores nos displays de 7 segmentos não foi realizada devido a problemas enfrentados durante o desenvolvimento.

Embora não tenhamos alcançado todos os resultados previstos inicialmente, conseguimos criar um jogo funcional e divertido para os usuários. Quanto aos botões, decidimos não incluir o botão de resetar o jogo, pois sua principal função seria reiniciar as pontuações e o tabuleiro, algo que não seria registrado devido à ausência dos displays.

É importante ressaltar o aprendizado adquirido com este projeto. Todos os membros da equipe ganharam experiência valiosa em programação, montagem de circuitos no Proteus e na placa perfurada. Uma habilidade destacada foi a soldagem, já que cada ligação, por menor que fosse, exigia duas soldas, e as conexões no PIC16F887, em particular, são muito próximas entre si.

Nosso objetivo, após a conclusão da disciplina, é continuar desenvolvendo este projeto para aprimorar nossas habilidades em montagem de circuitos e programação de componentes, buscando alcançar resultados ainda mais satisfatórios e completos.

REFERÊNCIAS

- STEPHEN. Jogo da Velha com Microcontrolador PIC e Eletrônica. MakerHero, 17 de agosto de 2018. Seção Embarcados, Projetos. Disponível em: https://www.makerhero.com/blog/jogo-da-velha-com-microcontrolador-pic/. Acesso em: Data não disponível.
- ALLDATASHEET, 2007. PIC16F887 Datasheet (PDF) - Microchip Technology. Disponível em: https://www.alldatasheet.com/datasheet-pdf/pdf/197543/MICROCHIP/PIC16F887.html. Acesso em: 26 de jun. de 2024.
- DUARTE, Y. D. S.; LOPES, J. F. A. Jogo da Velha Utilizando PIC. GitHub, 2024. Disponível em: https://github.com/JoseArantes83/Jogo_Da_Velha_Com_PIC16F887. Acesso em: 8 de jul. 2024.